**DAILY ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **24/06/2020** | **Name:** | **Dhavala** |
| **Course:** | **C++ Programming** | **USN:** | **4AL17EC027** |
| **Topic:** | * **Module 4: Classes and objects** | **Semester & Section:** | **6TH SEM & A Section** |
| **Github Repository:** | **Dhavala27** |  |  |

|  |
| --- |
| **SESSION DETAILS** |
|  |

|  |
| --- |
| **Report** What is an Object Object Oriented Programming is a programming style that is intended to make thinking about programming closer to thinking about the real world. In programming, objects are independent units, and each has its own identity, just as objects in the real world do.Objects An object might contain other objects but they're still different objects. Objects also have characteristics that are used to describe them. For example, a car can be red or blue, a mug can be full or empty, and so on. These characteristics are also called attributes. An attribute describes the current state of an object. Objects can have multiple attributes (the mug can be empty, red and large).What is a Class Objects are created using classes, which are actually the focal point of OOP. The class describes what the object will be, but is separate from the object itself. In other words, a class can be described as an object's blueprint, description, or definition. You can use the same class as a blueprint for creating multiple different objects. For example, in preparation to creating a new building, the architect creates a blueprint, which is used as a basis for actually building the structure. That same blueprint can be used to create multiple buildings. Programming works in the same fashion. We first define a class, which becomes the blueprint for creating objects. Each class has a name, and describes attributes and behavior. In programming, the term type is used to refer to a class name: We're creating an object of a particular type.Abstraction Data abstraction is the concept of providing only essential information to the outside world. It's a process of representing essential features without including implementation details. A good real-world example is a *book*: When you hear the term book, you don't know the exact specifics, i.e.: the page count, the color, the size, but you understand the idea of a book - the abstraction of the book.Encapsulation Part of the meaning of the word encapsulation is the idea of "surrounding" an entity, not just to keep what's inside together, but also to protect it. In object orientation, encapsulation means more than simply combining attributes and behavior together within a class; it also means restricting access to the inner workings of that class. The key principle here is that an object only reveals what the other application components require to effectively run the application. All else is kept out of view.Access Specifiers Access specifiers are used to set access levels to particular members of the class. The three levels of access specifiers are public, protected, and private. A public member is accessible from outside the class, and anywhere within the scope of the class object. For example:  #include <iostream> #include <string> using namespace std; class myClass { public: string name; }; int main() { myClass myObj; myObj.name = "SoloLearn"; cout << myObj.name; return 0; } //Outputs "SoloLearn" Private A private member cannot be accessed, or even viewed, from outside the class; it can be accessed only from within the class. A public member function may be used to access the private members. For example: #include <iostream> #include <string> using namespace std; class myClass { public: void setName(string x) { name = x; } private: string name; }; int main() { myClass myObj; myObj.setName("John"); return 0; } |

**DAILY ASSESSMENT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **24/06/2020** | **Name:** | **Dhavala** |
| **Course:** | **C++ Programming** | **USN:** | **4AL17EC027** |
| **Topic:** | * **Module 4: More on classes** | **Semester & Section:** | **6TH SEM & A Section** |
| **Github Repository:** | **Dhavala27** |  |  |

|  |
| --- |
| **SESSION DETAILS** |
|  |

|  |
| --- |
| **Report** Creating a New Class It is generally a good practice to define your new classes in separate files. This makes maintaining and reading the code easier. To do this, use the following steps in CodeBlocks: Click File->New->Class... Give your new class a name, uncheck "Has destructor" and check "Header and implementation file shall be in same folder", then click the "Create" button.Source & Header The header file (.h) holds the function declarations (prototypes) and variable declarations. It currently includes a template for our new MyClass class, with one default constructor. MyClass.h#ifndef MYCLASS\_H #define MYCLASS\_H class MyClass { public: MyClass(); protected: private: }; #endif // MYCLASS\_H The implementation of the class and its methods go into the source file (.cpp). Currently it includes just an empty constructor. MyClass.cpp#include "MyClass.h" MyClass::MyClass() { //ctor }Destructors Remember constructors? They're special member functions that are automatically called when an object is created. Destructors are special functions, as well. They're called when an object is destroyed or deleted.#ifndef & #define We created separate header and source files for our class, which resulted in this header file.#ifndef MYCLASS\_H #define MYCLASS\_H class MyClass { public: MyClass(); protected: private: }; #endif // MYCLASS\_H  ifndef stands for "if not defined". The first pair of statements tells the program to define the MyClass header file if it has not been defined already. endif ends the condition.Constant Objects As with the built-in data types, we can make class objects constant by using the const keyword.const MyClass obj; All const variables must be initialized when they're created. In the case of classes, this initialization is done via constructors. If a class is not initialized using a parameterized constructor, a public default constructor must be provided - if no public default constructor is provided, a compiler error will occur. Once a const class object has been initialized via the constructor, you cannot modify the object's member variables. This includes both directly making changes to public member variables and calling member functions that set the value of member variables.Composition In the real world, complex objects are typically built using smaller, simpler objects. For example, a car is assembled using a metal frame, an engine, tires, and a large number of other parts. This process is called composition. In C++, object composition involves using classes as member variables in other classes. This sample program demonstrates composition in action. It contains Person and Birthday classes, and each Person will have a Birthday object as its member. Birthday:class Birthday { public: Birthday(int m, int d, int y) : month(m), day(d), year(y) {  } private: int month; int day; int year; }; Our Birthday class has three member variables. It also has a constructor that initializes the members using a member initialization list. |